EIE
Escuela de
Ingeniería Eléctrica

Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

UNIVERSIDAD DE
COSTA RICA

IE-0117
Programación Bajo Plataformas Abiertas

Proyecto Final

Fernando Jiménez Ureña B74020
Delany Quirós Quesada B68285
Daniel Chacón Mora B72018

II ciclo
Diciembre 2021

1. Resumen

En el presente proyecto se busca crear un videojuego de tipo arcade basado en el famoso videojuego de la década de los 80s conocido como Pac-Man. Para poder programar el videojuego, se acudió a utilizar el lenguaje de programación Python.

El videojuego consiste en controlar un personaje dentro de un laberinto donde debe 'comer' todos los puntos repartidos en el mapa mientras es perseguido por otros cuatro personajes controlados por la computadora. A medida que el personaje 'come' los puntos, su puntuación va en aumento y va desbloqueando diferentes mapas.

2. Objetivos

2.1. Objetivo General

- Diseñar y construir un código de programación en Python para recrear e innovar el videojuego clásico de los 80, Pacman.

2.2. Objetivos específicos

- Crear un programa interactivo para el usuario.
- Adquirir nuevos conocimientos sobre la codificación y el desarrollo de videojuegos en 2D.
- Aplicar técnicas, conocimientos y herramientas útiles para videojuegos en 2D adquiridas en este curso para implementar nuevos personajes y 3 mapas distintos de los conocidos.

3. Alcances

Para el desarrollo de este proyecto se pretende alcanzar una experiencia de juego fluida para el usuario con el código desarrollado en Python aprovechando los conocimientos adquiridos durante este semestre en el curso de Programación Bajo Plataformas Abiertas. Se busca implementar una interfaz gráfica básica amigable con el usuario para que este pueda dirigir a su personaje en la huida de los monstruos que lo persiguen hasta que el juego se acabe.

4. Justificación

La programación de videojuegos presenta un desafío para las habilidades del programador donde se diseñará una imagen a partir de una serie de informaciones para trasladarla a la pantalla., así mismo con el sonido. Además procesará todas las interacciones de un agente con el entorno en base a una serie de entradas del usuario y por último asociar y procesar los fenómenos controlados por el usuario.

Los juegos acompañan el desarrollo humano porque, junto a otros hitos importantes, como el uso de imágenes y/o personajes para transmitir ideas y el desarrollo de estructuras conceptuales, como el arte, la ciencia, la cultura y la sociedad, nos han cambiado en nuestro presente. En los últimos años, los videojuegos han seguido desempeñando el mismo papel que los juegos, pero expandieron su potencial aprovechando las instalaciones tecnológicas que tenemos hoy.

El poder de los videojuegos para hacer que sus jugadores sientan las mismas emociones que sienten en la vida real, pero causadas por una realidad alterna o virtual, es lo que hace una herramienta poderosa, incluso más que otros medios físicos o electrónicos similares como la lectura, la música o las películas, por el simple hecho de permitir al jugador interactuar con dicha realidad alterna, algo no posible con los medios antes mencionados.

5. Marco Teórico

5.1. Videojuegos: Características Generales

Un videojuego es un software creado para el entretenimiento en general y en el que una o varias personan interactúan por medio de un controlador con un dispositivo que muestra la ejecución de dicho software. En la década de 1950 se dieron los primeros pasos en el desarrollo de videojuegos con la creación de títulos como *OXO* creado por AS Douglas como parte de su tesis doctoral. Durante estos años el desarrollo de videojuegos se daba más que todo para la investigación y como curiosidad informática por parte de los profesionales de la época.[2]

Fue hasta la década de 1970 que se empezó a dar un crecimiento exponencial en el desarrollo de videojuegos orientado más al entretenimiento y como un negocio con la aparición de consolas para la venta en el mercado. Desde entonces hasta el día de hoy, la industria de los videojuegos no ha dejado de crecer y lo que empezó como pura curiosidad por parte de los informáticos en 1950, hoy es una de las industrias del entretenimiento que más genera dinero superando incluso a la industria del cine con una facturación de 152.100 millones de dólares al año aproximadamente. [3]

Los videojuegos se pueden dividir dentro de diferentes 'géneros' que dependen de distintos factores como el sistema de juego, el tipo de interactividad con el jugador, sus objetivos, etc. Entre los principales géneros destacan Acción, Estrategia, Simulación, Deporte, Carreras y entre otros. Dentro de estos diferentes géneros existen los *Arcades*, los cuales destacan por

su jugabilidad simple, repetitiva y de acción rápida.

Originalmente estos videojuegos fueron creados para ser utilizados únicamente en máquinas 'Arcade' que consisten en máquinas dedicadas únicamente para correr este tipo de videojuegos. El uso de esta máquinas fue bastante popular en la década de los 70's y 80's y significó un modelo de negocio bastante lucrativo en estas épocas debido a que estas máquinas eran colocadas en lugares como centros comerciales, restaurantes, bares, o salones recreativos especializados y cobraban un monto de dinero al usuario para que pudiese jugar. Además fueron el inicio de una enorme rama en la computación que es la industria del desarrollo de videojuegos. [4]

Sin embargo, gracias al avance en la tecnología ahora es posible disfrutar este tipo de videojuegos en plataformas como celulares y computadores. En este caso se logró recrear uno de los más emblemáticos videojuegos de esta época como lo es Pac-Man utilizando recursos aprendidos en este curso y otros más desde diferentes tipos de fuentes.

5.2. Python

Para el desarrollo del videojuego, se utilizó el lenguaje de programación Python. Python lenguaje de programación de alto nivel, orientado a objetos, con una semántica dinámica integrada, principalmente para el desarrollo web y de aplicaciones informáticas. A diferencia de otros lenguajes de programación como el caso de C que se estudió en el curso, Python está más dirigido al Desarrollo Rápido de Aplicaciones debido a su tipificación dinámica y opciones de encuadernación dinámicas. Otra característica importante de Python es que es un lenguaje interpretado. Un lenguaje interpretado convierte el código escrito a lenguaje de máquina a medida que es ejecutado. [5]

Este dinamismo y su sintaxis centrada la legibilidad, permite que Python sea un lenguaje relativamente más simple e intuitivo. Esto facilita el desarrollo de software logrando reducir el costo de mantenimiento y de recursos ya que permite que los equipos trabajen en colaboración sin barreras significativas de lenguaje y experimentación.

Que un lenguaje de programación sea de alto nivel como Python, signfica que los algoritmos expresados mediante código, se adecuan de una manera más intuitiva a la capacidad cognitiva humana. Esto signfica que no trata con con registros, direcciones de memoria y las pilas de llamadas. Los lenguajes de programación de alto nivel manejan variables, matrices, objetos, aritmética compleja o expresiones booleanas, subrutinas y funciones, etc.[6]

Además de ser de alto nivel, Python también es un lenguaje de programación Orientado a Objetos. Esto significa que su funcionamiento se basa en Clases y Objetos. Esto permite estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos. El diseño de software en un lenguaje Orientado a Objetos gira en torno a datos u objetos, en lugar de funciones y lógica. [7]

Todas estas características ya mencionadas sobre Python han permitido el desarrollo de distintas librerías como es el caso de Pygame: Una librería dedicada para el desarrollo de videojuegos orientado al manejo de sprites, es decir, se enfoca en manejar imágenes simples; moverlas, alterarlas, duplicarlas, entre otras cosas.

5.3. Pygame

Pygame es una librería multiplataforma sobre SDL para Python; esta librería es utilizada en la implementación de juegos y aplicaciones multimedia en 2 dimensiones. Con sus clases y módulos brinda soporte al desarrollador para importar, tratar y exportar imágenes en varios formatos, IGC y formas básicos, efectos de sonido, reproducción de audio de fondo y CDs, reproducción de video MPEG, tratamiento de eventos de ratón, joystick, teclado, tiempo. [8]

5.3.1. Interfaz SDL

SDL es una librería GNU hecha para el tratamiento de gráficos y contenido multimedia, así como el control de dispositivos como video, teclado, mouse, unidad de CD/DVD, temporizador de la máquina, joystick, etc. Está fundamentalmente orientada a la implementación de juegos sencillos en 2D. [8]

5.3.2. Características de Pygame

- Generación de ventanas gráficas
- Manipulación y generación de eventos de dispositivos
- Tratamiento y exportación de imágenes desde y hacia ficheros de imagen de formatos conocidos como JPEG, PNG, GIF, TGA, BMP, entre otros
- Generación de formas, lineas y puntos básicos.
- Emisión de sonidos de efecto en ficheros OGG y WAV.
- Reproducción de música de fondo con archivos OGG, MP3 y MIDI.
- Sprites y control de colisiones.

5.3.3. Módulos principales de pygame

- pygame
- cd
- cursors
- draw
- event
- font
- image
- joystick
- key
- mask

- midi
- mixer
- mouse
- movie
- music
- scrap
- sprite
- time
- transform

6. Análisis de Resultados

6.1. Desarrollo del personaje principal controlado por el usuario

La definición de las direcciones de movimiento se logra por medio del uso de **vectores**. La posición en este juego 2D se representa por medio de las variables x y y. Es esta dirección vectorial la utilizada para describir la posicón y a la cuál constantemente se le está revisando su longitud por medio de la función *vector.py* en la sección de *def magnitude (self)*.

Para poder realizar la interacción con el usuario es fundamental determinar inicialmente la manera en la cuál el personaje se desplaza por el mapa. En el proyecto presente, se logra esto utilizando un método de *conexión de nodos* en donde los nodos son "puntos"del mapa donde se deben de trazar las rutas que pueden ser utilizadas por el usuario.

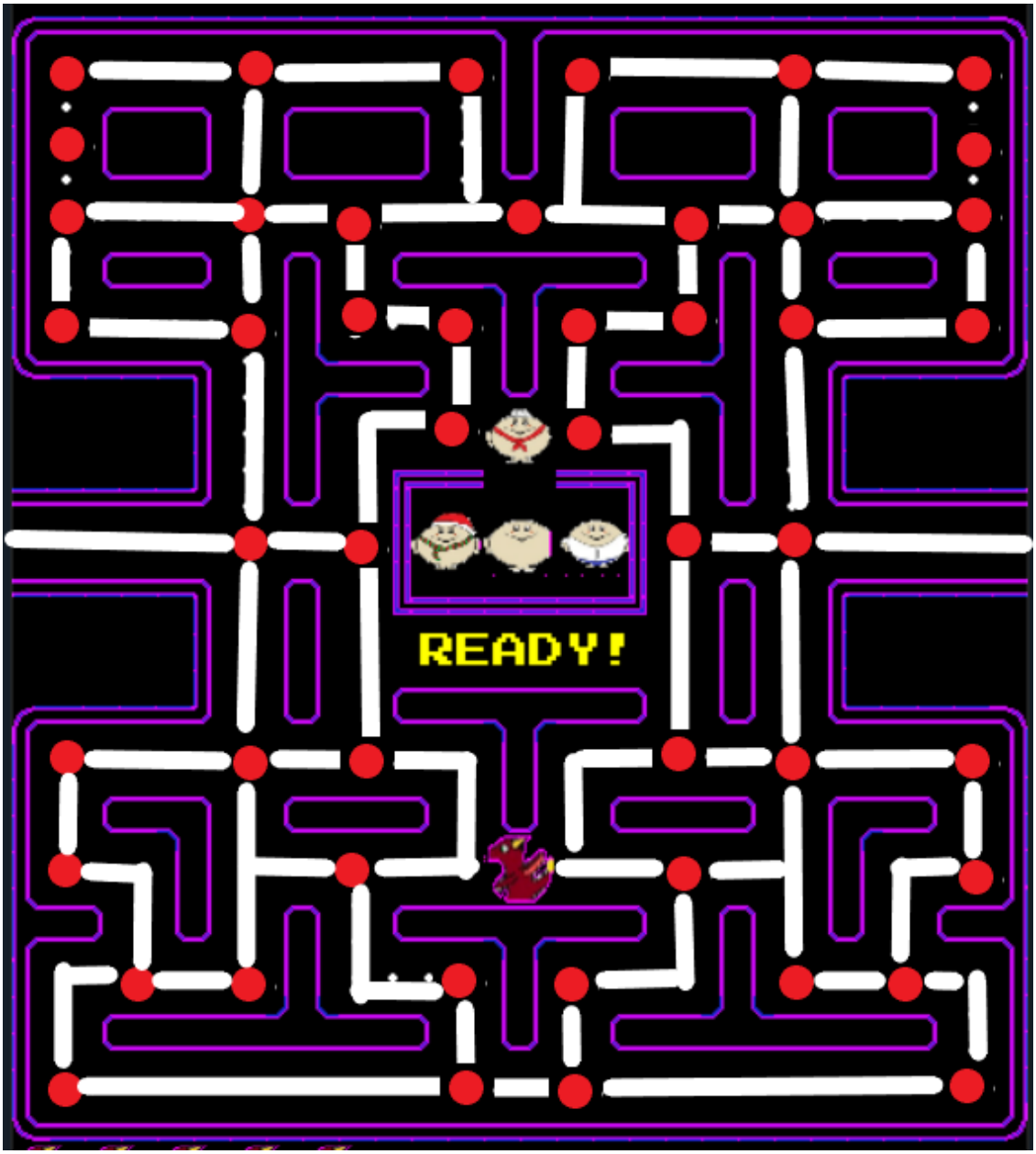

Figura 1: Mapa de notos interconectados en máximo 4 direcciones

estos nodos pueden tener como máximo 4 conexiones debido a las únicas cuatro direcciones de movimiento permitidas:

- Arriba ↑
- Abajo ↓
- Izquierda ⇐
- Derecha →

Para poder definir estos movimientos, se utiliza una manipulación aritmética de los movimientos ingresados en la función *vector.py*, ésta manipulación aritmética da como resultado un vector nuevo que es el que indica la dirección que posee en personaje en ese preciso instante del juego. Aquí cabe resaltar la manera utilizada por python para almacenar datos en la memoria debido a que para esta magnitud de vector cambiante en el tiempo es necesario realizar una copia la cuál va a ser modificada en cada interacción que se realice con el programa.

La función realizada encargada de los nodos se llama *nodes.py*, en ella se especifica cuáles rutas se pueden tomar utilizando la clase *nodegroup* debido a que es necesario agrupar todos los nodos presentes en el mapa para que quede la ruta correspondiente establecida.

```
class NodeGroup(object):
    def __init__(self, level):
        self.level = level
        self.nodesLUT = {}
        self.nodeSymbols = ['+', 'P', 'n']
        self.pathSymbols = ['.', '-', '|', 'p']
        data = self.readMazeFile(level)
        self.createNodeTable(data)
        self.connectHorizontally(data)
        self.connectVertically(data)
        self.homekey = None
```

Figura 2: Clase *nodegroup* utilizada para agrupar nodos

Es en esta misma función donde se le ordena a python renderizar"la ruta a seguir, la casa construida con nodos para las tortiricas y que el drago(personaje principal) no pueda ingresar a este "homenode".

```
def createHomeNodes(self, xoffset, yoffset):
    homedata = np.array([['X','X','+','X','X'],
                         ['X','X','.','X','X'],
                         ['+','X','.','X','+'],
                         ['+','.','+','.','+'],
                         ['+','X','X','X','+']])

    self.createNodeTable(homedata, xoffset, yoffset)
    self.connectHorizontally(homedata, xoffset, yoffset)
    self.connectVertically(homedata, xoffset, yoffset)
    self.homekey = self.constructKey(xoffset+2, yoffset)
    return self.homekey
```

Figura 3: Definición de nodos donde se encuentra la casa de las tortirricas

Definición de la casa por medio de nodos.

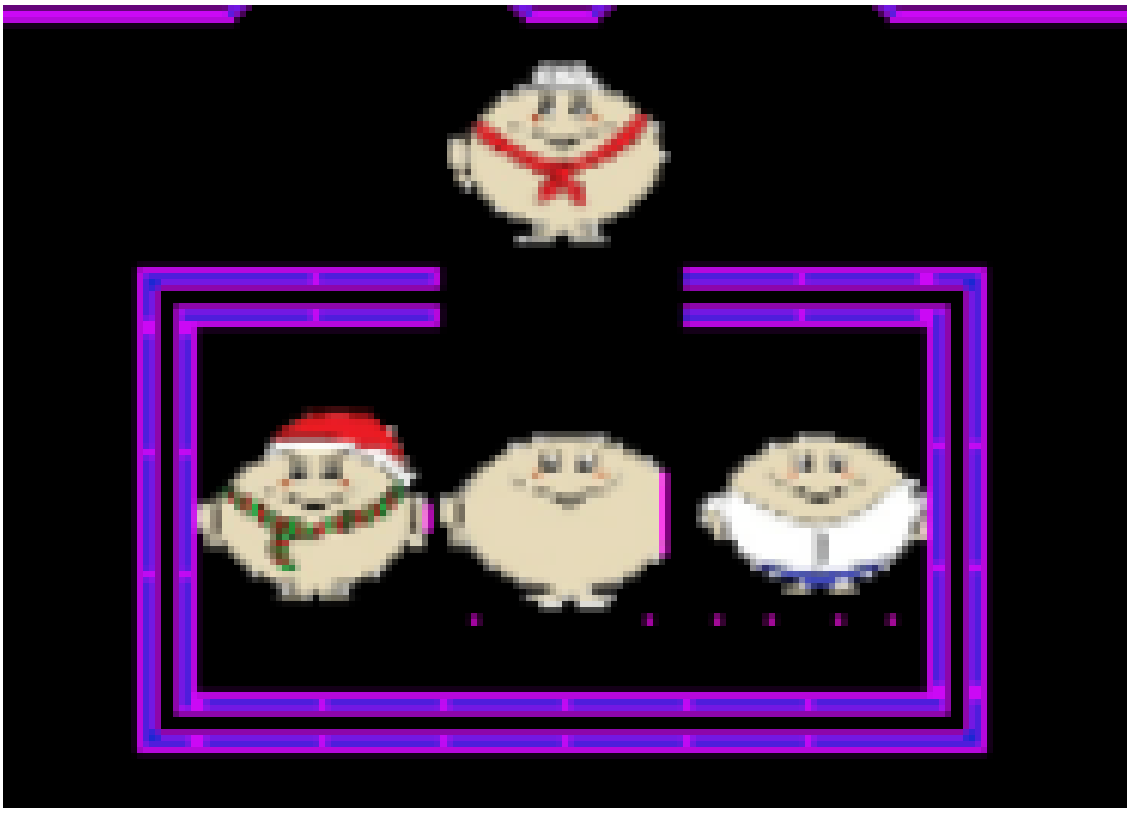

Figura 4: Casa de las tortirricas con el borde delimitado para evitar el ingreso de *Drago*

.

En la *Figura 4* se observa una atemorizante tortirica saliendo de la casa. En esta casa se

permite al salida y entrada de los enemigos una vez "muertos"pero no la entrada del personaje principal.

6.2. Desarrollo del entorno del videojuego

6.3. Creación del mapa inicial

Para crear el mapa donde tanto Draco como las malvadas Torticas interactúan, primero es necesario crear una matriz la cual definirá el mapa del juego, en este caso se añadió un archivo llamado maze.txt en el cual viene dibujada la matriz como se logra apreciar en la figura 5 y que se leerá en la sección de código en **StartGame** dentro de la función **run.py**. Cabe destacar que para la creación de la matriz se toma en cuenta la teoría de nodos antes mencionada.
Ahora bien dentro de la figura 5 se pueden apreciar distintas numeraciones y símbolos, lo importante es describir en el código que representa cada número o símbolo. Todos los números de la matriz contiene la posición de **sprite** lo cual se explicará en la siguiente sección, entonces para la demás simbología se explica a continuación:

- **"..º -":** Representa la posición donde habrá una bolita.
- **"p":** Representa la ruta donde habrá una bolita poderosa.
- **"P":** Representa el nodo donde habrá una bolita poderosa.
- **"n":** Representa la colocación de un nodo.
- **":** Representa la colocación de un camino.
- **"—":** Representa la colocación de un camino.

```
X X X X X X X X X X X X X X X X X X X X X X X X X X
0 1 1 1 1 1 1 1 1 1 1 1 1 7 8 1 1 1 1 1 1 1 1 1 1 1 0
1 + . . . . + . . . . . + 3 3 + . . . . . + . . . . + 1
1 . 2 3 3 2 . 2 3 3 3 2 . 3 3 . 2 3 3 3 2 . 2 3 3 2 . 1
1 p 3 X X 3 . 3 X X X 3 . 3 3 . 3 X X X 3 . 3 X X 3 p 1
1 . 2 3 3 2 . 2 3 3 3 2 . 2 2 . 2 3 3 3 2 . 2 3 3 2 . 1
1 + . . . . + . . + . . + . . + . . + . . + . . . . + 1
1 . 2 3 3 2 . 2 2 . 2 3 3 3 3 3 3 2 . 2 2 . 2 3 3 2 . 1
1 . 2 3 3 2 . 3 3 . 2 3 3 9 9 3 3 2 . 3 3 . 2 3 3 2 . 1
1 + . . . . + 3 3 + . . + 3 3 + . . + 3 3 + . . . . + 1
0 1 1 1 1 6 . 3 9 3 3 2 | 3 3 | 2 3 3 9 3 . 6 1 1 1 1 0
X X X X X 1 . 3 9 3 3 2 | 2 2 | 2 3 3 9 3 . 1 X X X X X
X X X X X 1 . 3 3 n - - n - - n - - n 3 3 . 1 X X X X X
X X X X X 1 . 3 3 | 4 5 5 = = 5 5 4 | 3 3 . 1 X X X X X
1 1 1 1 1 6 . 2 2 | 5 X X X X X X 5 | 2 2 . 6 1 1 1 1 1
n - - - - - + - - n 5 X X X X X X 5 n - - + - - - - - n
1 1 1 1 1 6 . 2 2 | 5 X X X X X X 5 | 2 2 . 6 1 1 1 1 1
X X X X X 1 . 3 3 | 4 5 5 5 5 5 5 4 | 3 3 . 1 X X X X X
X X X X X 1 . 3 3 n - - - - - - - - n 3 3 . 1 X X X X X
X X X X X 1 . 3 3 | 2 3 3 3 3 3 3 2 | 3 3 . 1 X X X X X
0 1 1 1 1 6 . 2 2 | 2 3 3 9 9 3 3 2 | 2 2 . 6 1 1 1 1 0
1 + . . . . + . . + . . + 3 3 + . . + . . + . . . . + 1
1 . 2 3 3 2 . 2 3 3 3 2 . 3 3 . 2 3 3 3 2 . 2 3 3 2 . 1
1 . 2 3 9 3 . 2 3 3 3 2 . 2 2 . 2 3 3 3 2 . 3 9 3 2 . 1
1 P . + 3 3 + . . + . . + - - + . . + . . + 3 3 + . P 1
8 3 2 . 3 3 . 2 2 . 2 3 3 3 3 3 3 2 . 2 2 . 3 3 . 2 3 7
7 3 2 . 2 2 . 3 3 . 2 3 3 9 9 3 3 2 . 3 3 . 2 2 . 2 3 8
1 + . + . . + 3 3 + . . + 3 3 + . . + 3 3 + . . + . + 1
1 . 2 3 3 3 3 9 9 3 3 2 . 3 3 . 2 3 3 9 9 3 3 3 3 2 . 1
1 . 2 3 3 3 3 3 3 3 3 2 . 2 2 . 2 3 3 3 3 3 3 3 2 . 1
1 + . . . . . + . . + . . . . . + 1
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
X X X X X X X X X X X X X X X X X X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X X X X X X X
```

Figura 5: Matriz en .txt para el juego

6.4. Mapa Gráfico

Como se mencionó anteriormente los números dentro de la matriz representan los **sprites** del mapa, ahora los sprites son básicamente una imagen que se colocará en la posición del número que se le asigne dentro de la matriz, para este proyecto se utilizaron 3 juegos de 6 sprites de diferente color para la creación del mapa 6, los cuales pueden girarse en los grados necesarios por ejemplo para crear las 4 esquinas del mapa. Es importante mencionar que en un sprite al estar todas las imágenes en él la sección especifíca del sprite se toma como un cuadrado tomando sus coordenadas en **x** y **y** para determinar su ubicación. Una vez que el mapa quede con un diseño exitoso y sin problemas se pueden apagar los nodos.

Figura 6: Sprites para la creación del mapa

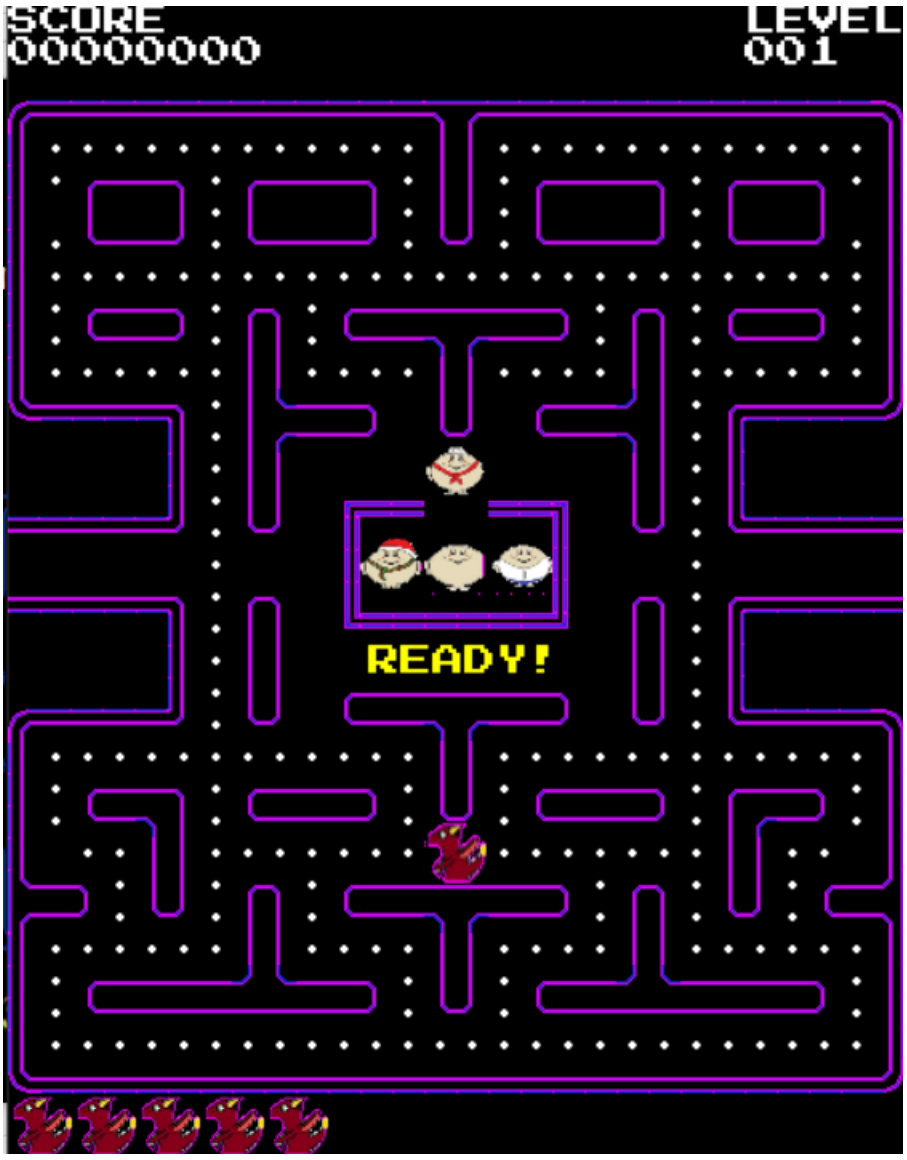

Figura 7: Mapa 1

6.5. Desarrollo de los personajes controlados por la computadora

El desarrollo de otros personajes que persigan al personaje principal alrededor del mapa es la parte más compleja del programa. Sin embargo, gran parte del código que ya fue desarrollado

para el movimiento de Draco de nodo a nodo será utilizado en el desarrollo de estos diferentes personajes.

Un punto bastante importante que difiere el movimiento de estos personajes con respecto al movimiento de Draco, es que estos no pueden moverse en reversa. Ellos eligen su dirección de manera automática dependiendo del destino al cual desean llegar.

Sin embargo, es un hecho que tanto los enemigos controlados por la computadora como Draco, se mueven por el mapa. Por lo tanto se procedió a crear un clase llamada *Entity*. Dentro de esta clase se añaden los movimientos de Draco ya explicados anteriormente y además los movimientos de los enemigos así como los objetos que van apareciendo en el mapa.

Dentro de la función *update()*, los enemigos tendrán un comportamiento muy similar al programado para Draco, sin embargo cada vez que ellos llegan a un nuevo nodo, su dirección es elegida aleatoriamente y no por el usuario. Básicamente esto se logra gracias a la función validDirections(), que depende de la posición en que se encuentre en ese momento la TortiRica, entonces pasa a la función *update()* la posibles direcciones que puede tomar y finalmente gracias al método *randint*, entonces aleatoriamente el programa elige una de esas direcciones posibles y se mueve al siguiente nodo.

```
def validDirections(self):
    directions = []
    for key in [UP, DOWN, LEFT, RIGHT]:
        if self.validDirection(key):
            if key != self.direction * -1:
                directions.append(key)
    if len(directions) == 0:
        directions.append(self.direction * -1)
    return directions

def randomDirection(self, directions):
    return directions[randint(0, len(directions)-1)]
```

Figura 8: Funcion ValidDirections() y randomDirection()

Ya una vez definidas las funciones que se encargan del movimiento de los personajes(Tortiricas), se procede a crear una clase llamada *Ghosts* para ya poder colocar un personaje dentro del mapa al mismo tiempo que se está controlando al personaje principal(Draco). Para este momento, únicamente se ha logrado crear un personaje que se está moviendo de manera aleatoria por el mapa sin ningún objetivo en concreto.

Por lo tanto, se procede a darle un cierto tipo de inteligencia para que no únicamente ande de manera aleatoria por el mapa, si no que tenga un objetivo cada cierto tiempo para alcanzar. Este 'objetivo' consiste en definir cada cierto tiempo un vector (x,y) al cual el personaje(Tortirica) debe llegar. Este código se implementa dentro de la clase *Ghost()*.

```
def scatter(self):
    self.goal = Vector2(TILEWIDTH*NCOLS, TILEHEIGHT*NROWS)
```

Figura 9: Vector que define la dirección de la Tortirica

A las TortiRicas se les programan cuatro diferentes modos para simular que poseen cierto tipo de 'inteligencia artificial'. El primer modo se le conoce como *Scatter*. Cuando las TortiRicas se encuentran bajo este modo, el programa les da un punto aleatorio en el mapa al cual deben llegar siguiendo la ruta más cercana por un lapso de 7 segundos. Este punto se genera gracias a los métodos implementados y descritos anteriormente con los vectores.

Luego después del lapso de 7 segundos, el programa cambia el modo en el que se encuentra la TortiRicas a *Chase*. Cuando se encuentran en este modo, el programa en vez de generar un punto aleatorio en donde la TortiRica debe llegar, le brinda las coordenadas exactas de donde se encuentra Draco en ese momento. Por lo tanto lo persigue por un lapso de 20 segundos y luego pasado este tiempo, entonces la TortiRica vuelve de nuevo al modo *Scatter*.

Existen luego dos modos especiales en los que la TortiRica puede estar: *Freight* y *Spawn*. El primero, se activa en el momento en el que Draco come una de las bolitas especiales. Cuando las TortiRicas se encuentran en modo *Freight*, entonces se vuelven más lentas y Draco puede comerlas. En el momento que Draco las come, entonces entran al último modo llamado *Spawn*. En este modo, las Tortillas se mueven rápidamente de vuelta a la base y una vez en la base, vuelven a su modo ya sea *Scatter* o *Chase*

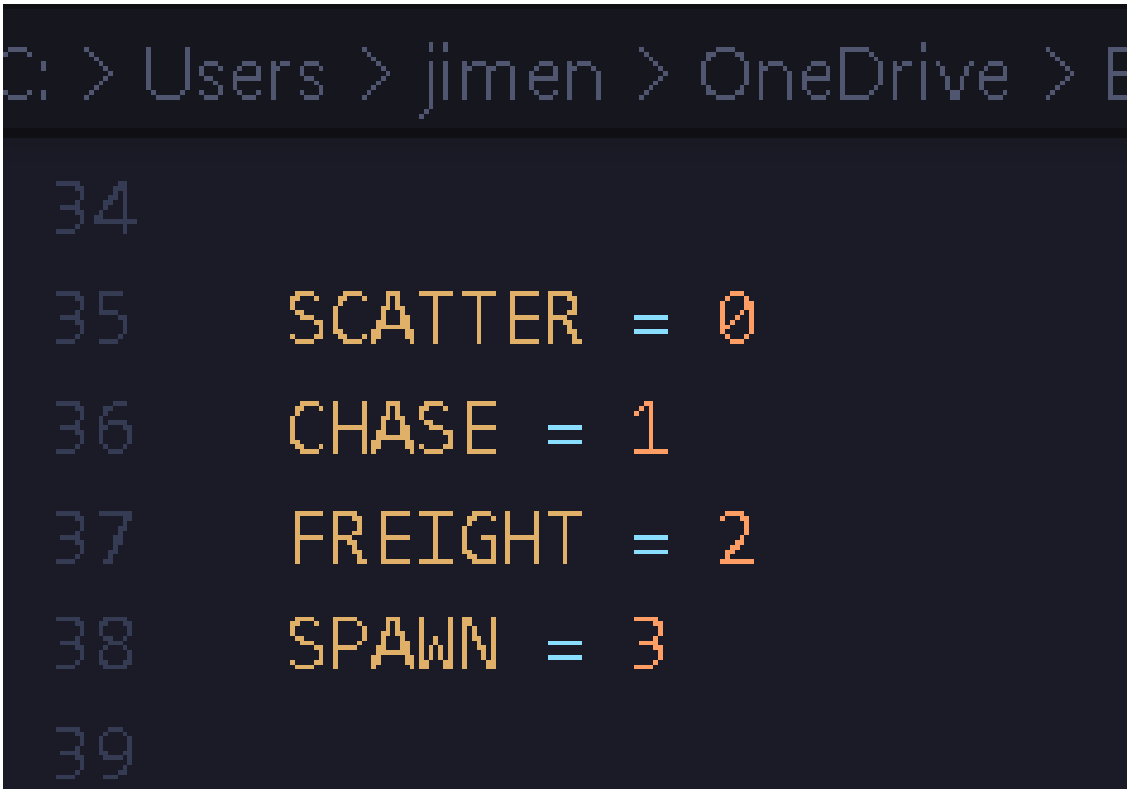

Figura 10: Constantes definidas para los modos en constants.py

```
class MainMode(object):
    def __init__(self):
        self.timer = 0
        self.scatter()

    def update(self, dt):
        self.timer += dt
        if self.timer >= self.time:
            if self.mode is SCATTER:
                self.chase()
            elif self.mode is CHASE:
                self.scatter()

    def scatter(self):
        self.mode = SCATTER
        self.time = 7
        self.timer = 0

    def chase(self):
        self.mode = CHASE
        self.time = 20
        self.timer = 0
```

Figura 11: Modos *Scatter* y *Chase* en la clase modes.py

```
def setFreightMode(self):
    if self.current in [SCATTER, CHASE]:
        self.timer = 0
        self.time = 7
        self.current = FREIGHT
    elif self.current is FREIGHT:
        self.timer = 0

def setSpawnMode(self):
    if self.current is FREIGHT:
        self.current = SPAWN
```

Figura 12: Modos *Freight* y *Spawn* en la clase modes.py

6.6. Desarrollo de elementos visuales y de audio del videojuego

Para los elementos visuales, se siguió utilizando los sprites, ahí se encontraron las distintas imágenes donde se debía agregar la imagen para un momento específico, como lo es la muerte de Draco, en la figura **??** se pueden apreciar los sprites utilizados para dicha animación. Lo que se debe hacer es ir a la clase Pacman y actualizar sus sprites en el método de actualización.

Por ejemplo, el método de actualización recorre la lista de fotogramas a una velocidad establecida previamente y devuelve cualquier marco que necesite devolver. Funciona con el método nextFrame que solo incrementa el contador current_frame cuando es necesario.

Figura 13: Animación de la muerte de Draco

De manera opcional se puede agregar música de fondo al juego, en este caso se creó una función llamada **music.py** que contiene unas líneas de código muy sencillas para agregar un sonido de fondo al juego.

```
import pygame
pygame.mixer.init()
pygame.mixer.music.load("C:/Users/Usuario/Downloads/Pacman_Complete/Pacman_Complete/pacman.mp3")
pygame.mixer.music.play(-1)
pygame.mixer.music.set_volume(0.1)
```

Figura 14: Función para añadir música

7. Conclusiones, observaciones y recomendaciones

La creación de matrices, su edición e inserción de elementos nuevos resulta ser una herramienta muy poderosa no solo para la manipulación de una gran cantidad de datos sino también para la descripción del cambio de posición con respecto al tiempo de algún elemento en un entorno delimitado.

Las aplicaciones para esto son innumerables, además la manera en la que los nuevos vectores con las direcciones son generados permite mantener un control de los movimientos hechos anteriormente y así tener un respaldo en caso de que se necesite para llevar un registro de estos movimientos.

La aplicación de un programa que se encarga de registrar las direcciones de los movimientos como desarrollo posterior a este trabajo podría resultar en una herramienta muy útil en la industria para la manipulación de objetos, como por ejemplo una cinta transportadora que tenga como función cambiar al dirección de los objetos que se encuentran en ella por medio de la introducción de nuevas direcciones específicas en una arreglo de datos grandes, como las matrices utilizadas en el presente proyecto y las utilizadas anteriormente en el curso.

En la creación de mapas existe la opción de usar tiles en vez de sprites, los tiles son partes de la imagen completa recortadas y divididas en pequeñas secciones, dependiendo de como se puede resultar más fácil el tiles para cargar la imagen que presentar la ubicación dentro del sprite.

Referencias

[1] Creación propia.

[2] Herrera, A. (2021). Historia de los Videojuegos. Retrieved 8 December 2021, from https://www.fib.upc.edu/retro-informatica/historia/videojocs.html.

[3] El videojuego en el Mundo - Asociación Española de Videojuegos. Asociación Española de Videojuegos. (2021). Retrieved 6 December 2021, from http://www.aevi.org.es/la-industria-del-videojuego/en-el-mundo/.

[4] Trenta, M. (2012). Orígenes del videojuego: conexiones históricas y sociales de un producto cultural. In Presentado en IV Congreso Internacional Lafina de Comunicación, Laguna.

[5] Python: qué es, para qué sirve y cómo se programa — Informática Industrial. aula21 — Formación para la Industria. (2021). Retrieved 9 December 2021, from https://www.cursosaula21.com/que-es-python/.

[6] González, H. (2021). ¿Qué es un lenguaje de programación de alto nivel?. Retrieved 8 December 2021, from https://herschelgonzalez.com/que-es-un-lenguaje-de-programacion-de-alto-nivel/.

[7] Osores, M. (2021). ¿Qué es Programación orientada a objetos, OOP?. ComputerWeekly.es. Retrieved 9 December 2021, from https://www.computerweekly.com/es/definicion/Programacion-orientada-a-objetos-OOP.

[8] EcuRed.(2021). Python Game Library. Retrieved 8 December 2021, from https://www.ecured.cu/Python_Game_Library

8. Anexos

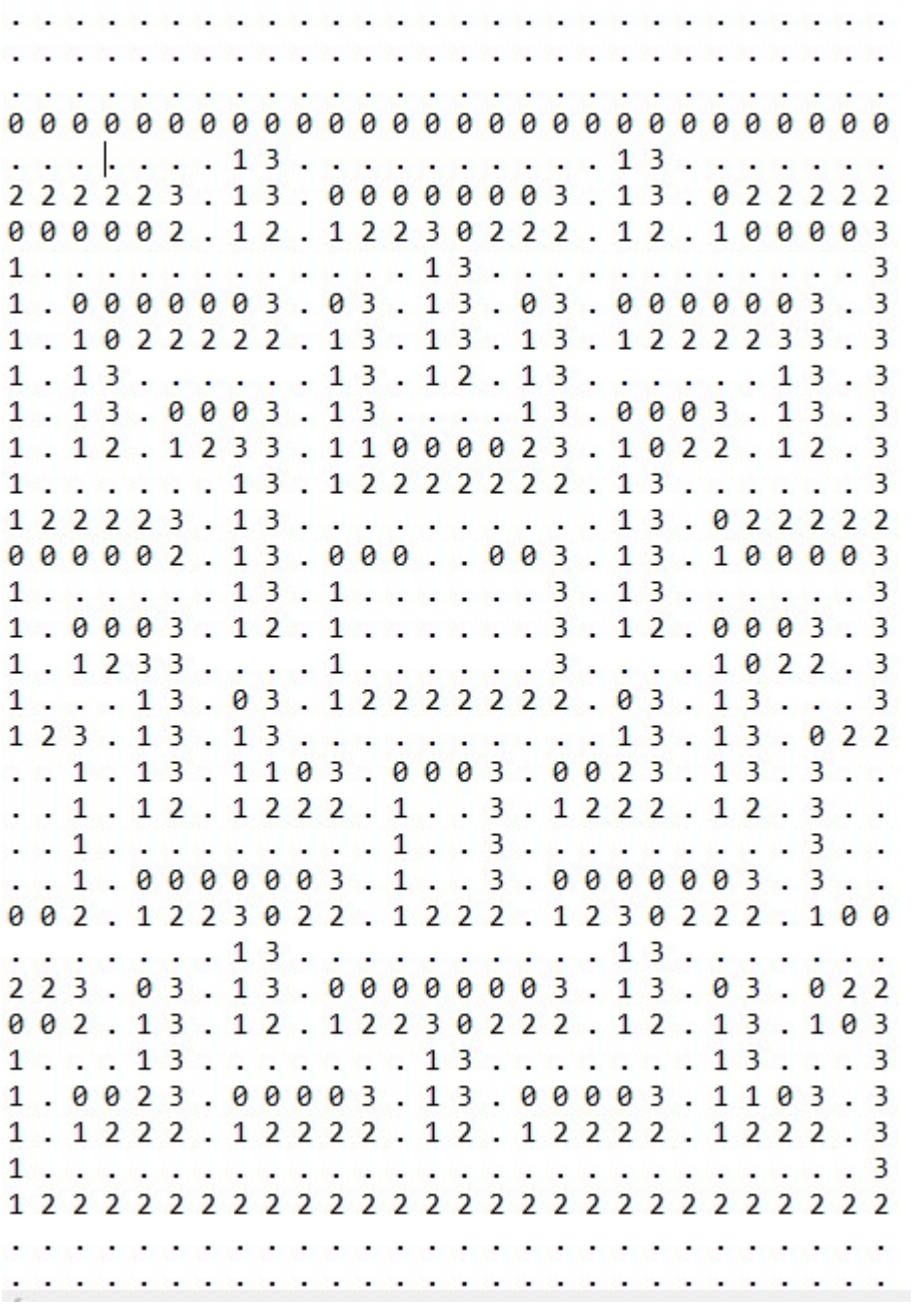

Figura 15: Matriz mapa 2

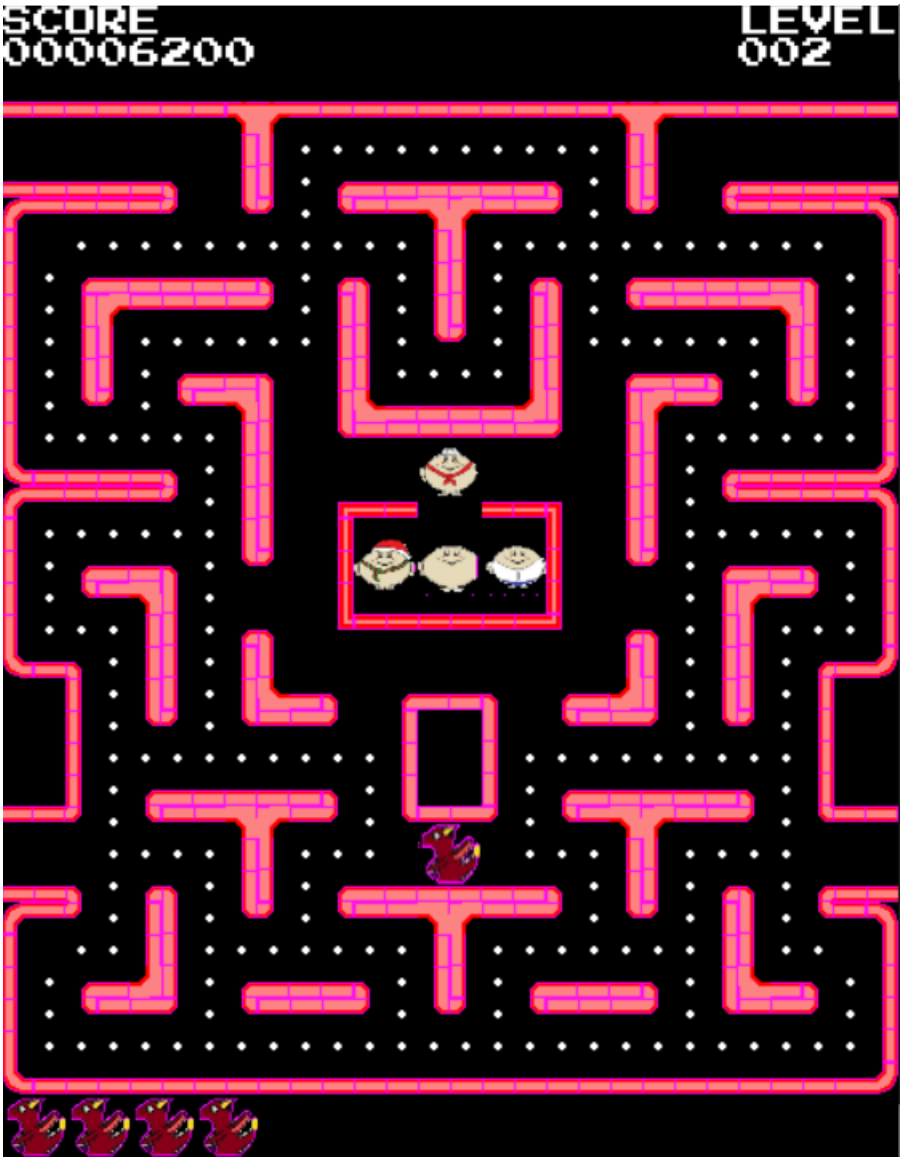

Figura 16: Mapa 2

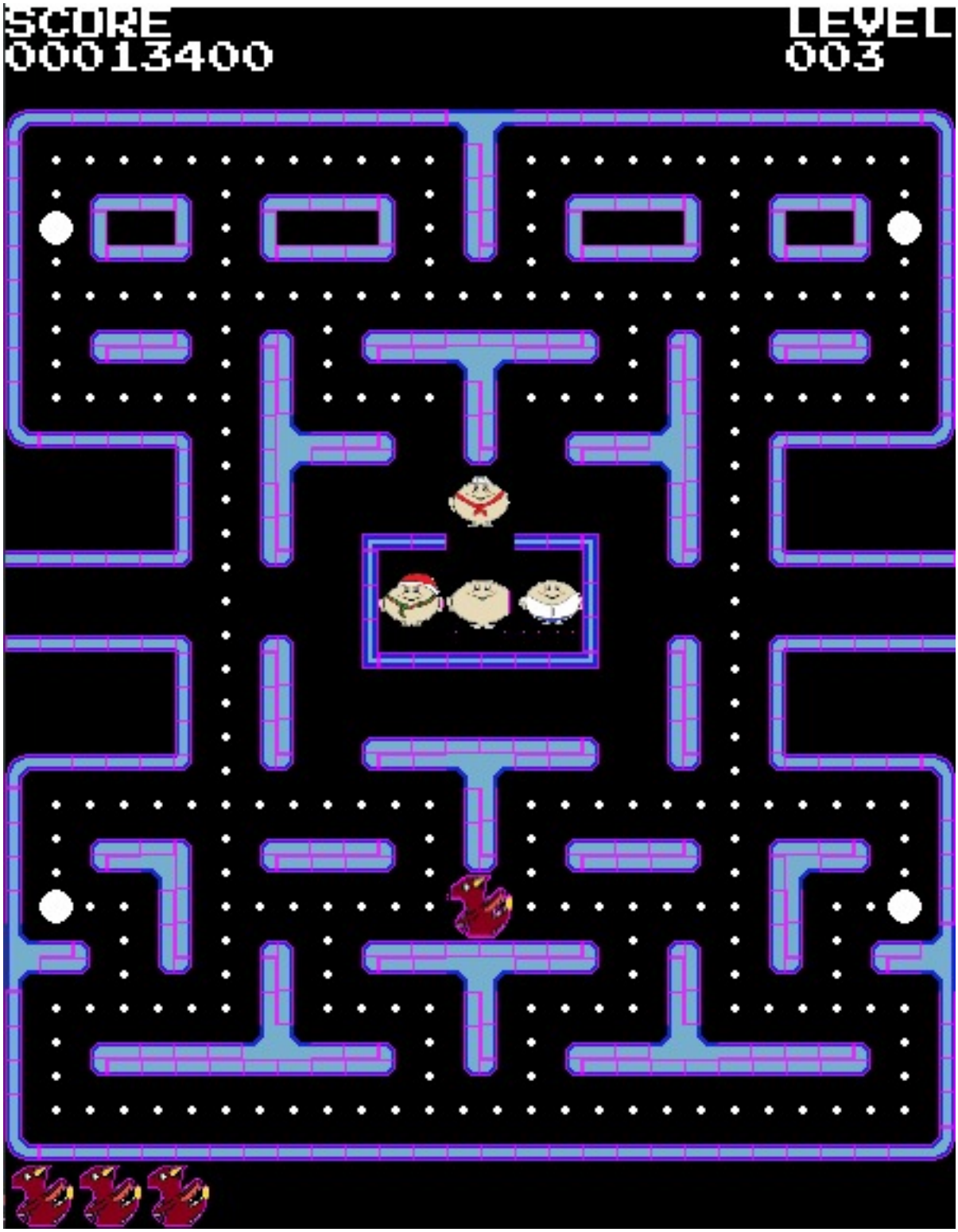

Figura 17: Mapa

Figura 18: Posiciones de Draco

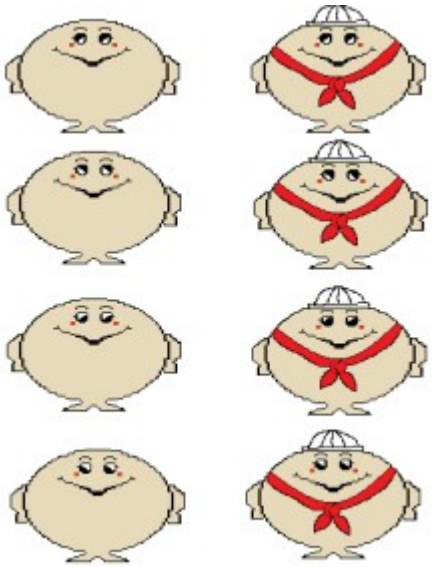

Figura 19: Diseño de las tortiricas 1 y 2

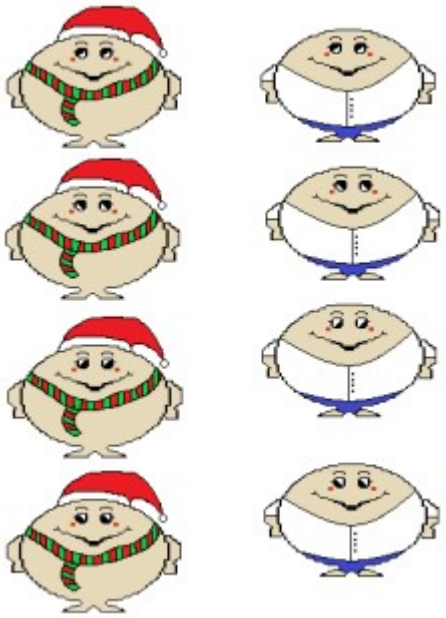

Figura 20: Diseño de las tortiricas 3 y 4

Figura 21: Diseño de la pachita, elemento especial equivalente a la fruta en pacman